3Л

УДК 004.8

Р.І. Лупійчук
Київський національний університет імені Тараса Шевченка, Україна
lj@univ.kiev.ua

Розрішення анафор: сучасні алгоритми

У статті розглядаються алгоритми розрішення займенникових анафор. Наводиться порівняльний аналіз розглянутих алгоритмів та продемонстровано їх роботу на конкретному прикладі. До розгляду взяті наступні алгоритми: Лапіна і Ліса, Хобса, центруючий та Міткова.

Вступ

Розрішення анафор залишається актуальною задачею комп'ютерної лінгвістики. Дослідники в цій сфері звітують про те, що розроблені ними алгоритми дозволяють знаходити антецеденти для займенників з 86%-ю і більше точністю [1]. Ця точність хоча і є високою, але не є достатньою, особливо зважаючи на те, що дослідження проводяться на тематичних корпусах текстів, зі спрощеною, офіційною, мовою.

У даному документі розглянуто на прикладі деякі найбільш відомі та визнані методи визначення антецедентів для займенникових анафор. Цей вид анафор є найбільш простим та дослідженим, тим не менше приклади є досить демонстративними.

Метою даної роботи є огляд сучасних методів розрішення займенникових анафор та розгляд їх на конкретному прикладі.

Алгоритм Лапіна і Ліса

Алгоритм використовує просту схему оцінки, яка задіює фактори близькості і граматичної ролі; семантичні фактори не враховуються, окрім тих, що задіюються при узгодженні. Кожна іменна група оцінюється за такими критеріями: близькість речення (100), підметниковий наголос (80), наголос існування (70), пряме доповнення (50), непряме доповнення (40), відмежена обставина (50), головний іменник (80). В дужках вказано кількість балів, які нараховуються відповідному можливому антецеденту.

Приклад

Lesya found some flax in a store. She said that to Ola. She bought it. (*)

Знаходимо можливі референти та їхні оцінки для першого речення.

Таблиця 1

	Rec	Subj	Exist	Obj	Ind-Obj	Non-Adv	Head N	Total
Lesya	100	80				50	80	310
flax	100			50			80	280
store	100						80	230

В цьому реченні немає займенників, тому ми переходимо до обробки наступного речення (табл. 2).

Таблиця 2

Референт	Фрази	Оцінка
Lesya	{ *Lesya* }	155
flax	{ *some flex* }	140
store	{ *a store* }	115

Колонка «фрази» містить клас еквівалентності, що містить посилання на референта.

Перша іменна група в другому реченні є займенник *she*. Так як *she* жіночого роду, на другому кроці кількість можливих референтів зменшується до одного Lesya, тому ми просто приймаємо його як референта.

Тепер необхідно оновити модель тексту. Насамперед, необхідно додати займенник *she* в клас еквівалентності Lesya, а потім змінити оцінку. Так як *she* знаходиться в поточному реченні, а *Lesya* в попередньому, то оцінку непотрібно ділити навпіл. Порахуємо оцінку для *she*: він знаходиться в поточному реченні (= 100), є підметом (= 80), не є допоміжним (adverbial) (= 50), не є вбудованим (embedded) (= 80), таким чином до оцінки Lesya треба додати 310. В результаті отримаємо:

Таблиця 3

Референт	Фрази	Оцінка
Lesya	{ *Lesya*, she_1 }	465
flax	{ *some flex* }	140
store	{ *a store* }	115

Наступна іменна група є *Ola*, яка є новим референтом. Для неї оцінка буде наступна: 100 + 40 + 50 + 80 = 270.

Таблиця 4

Референт	Фрази	Оцінка
Lesya	{ *Lesya*, she_1 }	465
Ola	{ *Ola* }	270
flax	{ *some flex* }	140
store	{ *a store* }	115

Переходимо до останнього речення. Знову зменшуємо оцінку вдвічі.

Таблиця 5

Референт	Фрази	Оцінка
Lesya	{ *Lesya*, she_1 }	232,5
Ola	{ *Ola* }	135
flax	{ *some flex* }	70
store	{ *a store* }	57,5

Таким самим чином переконуємось, що для she референтом буде Lesya, а для it референтом буде flax.

Алгоритм Хобса

В алгоритмі використовуються синтаксичні представлення речень, які вважаються даними. На відміну від алгоритму Лапіна і Ліса цей алгоритм не передбачає наявності моделі чи налаштувань дискурсу, хоча деякі налаштування враховуються способом обходу синтаксичних дерев.

Алгоритм

1. Починаємо з іменної групи (NP), яка містить займенник.

2. Піднімаємось по дереву до першої NP або речення (S). Назвемо цей вузол X і назвемо шлях до нього p.

3. Обходимо зліва-на-право, в ширину, усі гілки, що знаходяться нижче X та лівіше p. Візьмемо за антецедент будь-яку NP, що має вузол NP або S між ним і X.

4. Якщо вузол X є найвищим S в реченні, то обійти дерева попередніх речень в порядку їх появи, кожне дерево обходимо зліва-на-право, в ширину. Приймемо антецедентом перший NP вузол, що трапиться. Якщо X не є найвищим S вузлом, переходимо до кроку 5.

5. Переходимо вверх від вузла X, до першого NP або S вузла, що зустрінеться. Назвемо цей новий вузол X, а шлях до нього назвемо p.

6. Якщо X є NP вузлом і шлях p до X не проходить через іменну групу, в яку X безпосередньо входить, прийняти X антецедентом.

7. Обійти усі гілки нижче та лівіше вузла X, зліва-на-право, в ширину. Запропонувати будь-який NP вузол, що зустрінеться, як антецедент.

8. Якщо X є S вузлом, обійти усі гілки вузла X, зліва-на-право, в ширину, але не обходити підгілки NP та S вузлів, що будуть зустрічатися. Запропонувати будь-який NP вузол як антецедент.

9. Перейти до кроку 4 [2].

Розглянемо, як працює алгоритм Хобса, на прикладі (*).

Приклад

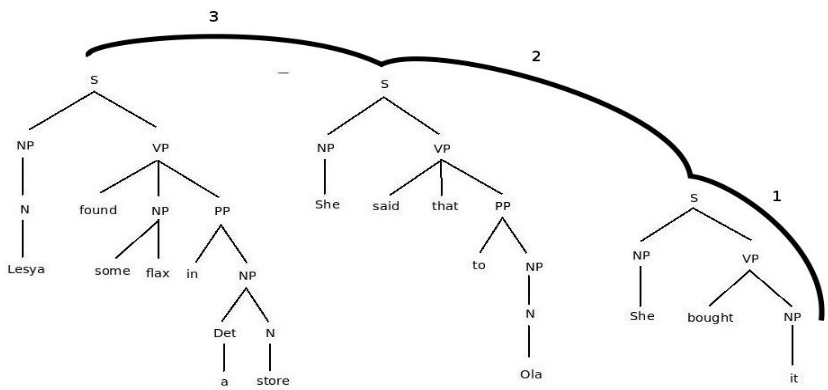


Рисунок 1 – Синтаксичне дерево для прикладу (*)

Знайдемо антецедент для займенника it в останньому реченні. Відповідно до кроку 2 піднімаємося вище по дереву до S, так як інших вузлів NP немає. Переконуємось, що немає вузлів NP, розміщених нижче S та лівіше від шляху, таких, що містять вузол NP між ними та S. Далі згідно з кроком 4 обходимо друге речення зліва-на-право, в ширину, і не зустрічаємо жодного NP, який узгоджується з it, тому переходимо до першого речення. Обходячи таким самим чином, приймаємо антецедентом для it іменну групу flax.

Центруючий алгоритм

Центруючий алгоритм, як і алгоритм Лапіна і Ліса, має чітку модель дискурсу, але намагається дати відповідь на додаткове питання: на чому одному «сконцентрований» дискурс в кожен конкретний момент.

Є два основних поняття, що використовуються в моделі дискурсу. Нехай U_n і U_{n+1} – два послідовні вирази. Зворотний центр (backward looking center) U_n, позначається $C_b(U_n)$, відображає сутність, на якій зосереджений дискурс, після того, як U_n було інтерпретовано. Прямі центри (forward looking centers) U_n, позначаються $C_f(U_n)$, утворюють упорядкований список сутностей, згаданих в U_n, які можуть послужити як C_b для наступного виразу. Фактично $C_b(U_{n+1})$ є за означенням елемент з $C_f(U_n)$ з найбільшою оцінкою, згаданий в U_{n+1}.

Робота алгоритму базується на відношенні $C_b(U_{n+1})$, $C_b(U_n)$, $C_p(U_{n+1})$, і показана в табл. 6.

Таблиця 6

	$C_b(U_{n+1})=C_b(U_n)$ or undefined $C_b(U_n)$	$C_b(U_{n+1}) \neq C_b(U_n)$
$C_b(U_{n+1})=C_p(U_{n+1})$	Продовжити	Гладкий зсув (Smooth-Shift)
$C_b(U_{n+1}) \neq C_p(U_{n+1})$	Зберегти	Брудний зсув (Rough-Shift)

Правила, що діють в алгоритмі:

1. Правило 1: Якщо будь-який елемент $C_f(U_n)$ поданий як займенник в реченні U_{n+1}, тоді $C_b(U_{n+1})$ має бути поданий займенником також.
2. Правило 2: Стани переходів впорядковані в наступному порядку: продовжити, зберегти, гладкий зсув, брудний зсув.

Алгоритм

1. Згенерувати можливі комбінації $C_b - C_f$ для кожного можливого набору антецедентів.
2. Відкинути ті, що не задовольняють обмеженням: синтаксичним, вибірковим, центруючим правилам і обмеженням.
3. Впорядкувати за перехідними відношеннями.

Приклад

Lesya found some flax in a store. (U_1)

She said that to Ola. (U_2)

She bought it. (U_3)

Використовуючи граматичну ієрархію для впорядкування C_f, для речення U_1 отримаємо:

$C_f(U_1)$: {Lesya, flax, store}

$C_p(U_1)$: Lesya

$C_b(U_1)$: undefinded

Речення U_2 містить займенник *She*. Він може бути зіставлений тільки з Lesya, через обмеження за родом. Lesya є за означенням $C_b(U_2)$, так як має найбільшу оцінку серед елементів $C_f(U_1)$, згаданих в U_2.

$C_f(U_2)$: {Lesya, Ola}

$C_p(U_2)$: Lesya

$C_b(U_2)$: Lesya

Переходимо до речення U_3.

$C_f(U_3)$: {Lesya, it}

$C_p(U_3)$: Lesya

$C_b(U_3)$: Lesya

Результат: Продовжити ($C_p(U_3)=C_b(U_3)=C_b(U_2)$)

$C_f(U_3)$: {Ola, it}

$C_p(U_3)$: Ola

$C_b(U_3)$: Lesya

Результат: Зберегти ($C_p(U_3) \neq C_b(U_3)=C_b(U_2)$)

Так як результат «Продовжити» є більш пріоритетним, ніж «Зберегти» за правилом 2 Lesya приймається антецедентом.

Алгоритм Міткова

Алгоритм Міткова дуже схожий на алгоритм Лапіна і Ліса. Схема роботи алгоритму така ж, але можливі кандидати в антецеденти оцінюються за кожним з критеріїв по шкалі від –1 до 2, самі критерії також відрізняються. Варто відмітити, що в даному алгоритмі антецедентом може бути обраний займенник, що дещо розширює можливості алгоритму.

Таблиця 7 – Критерії, що використовуються в алгоритмі Міткова

Критерій	Кількість очок
Синтаксичний паралелізм (Syntactic Parallelism)	1
Повторюваність кандидата (Frequent Candidates)	1
Схоже положення (Collocation Match)	2
Підмет (Subject Role)	2
Доповнення (Object Role)	1
Граматична роль не визначенна	–1
Часте згадування (Term Preference)	1

Повторюваність кандидата: цей індикатор надає додаткові очки (+1) трьом найбільш частим можливим кандидатам в антецеденти.

Схоже положення: цей індикатор додає +2, якщо займенник і можливий антецедент зустрічались в однакових контекстах в сенсі дієслова, що поряд.

Часте згадування: обирається 10 значущих слів, що мають найбільше TF.IDF. Усі антецеденти, в які входить принаймні одне з обраних слів, отримують додатковий бонус.

Покроковий алгоритм практично такий самий, як і у Лапіна та Ліса, за відмінністю того, що не враховуються очки, набрані в попередніх реченнях. Тобто для займенникової анафори виділяються можливі антецеденти, фільтруються за синтаксичними обмеженнями, потім оцінюються за кожним з критеріїв, оцінки сумуються, та обирається найближчий кандидат з максимальною оцінкою.

Висновки

Було розглянуто на конкретному прикладі деякі методи розрішення займенникових анафор. До розгляду ввійшло 4 методи: Лапіна і Ліса, Хобса, центруючий та Міткова. Деякі алгоритми, такі як центруючий алгоритм та Міткова, багато в чому повторюють метод Лапіна і Ліса.

Усі алгоритми знаходження антецедента так чи інакше враховують деякі обмеження, що накладаються на можливих кандидатів, такі як: узгодженість в числі (number agreement), особі (person agreement), відмінку (case agreement), роді (gender agreement); синтаксичні обмеження та локальні семантичні обмеження.

Алгоритм Лапіна і Ліса та Міткова працюють методом зважування кожного з антецедентів за переліком критеріїв, деякі з них контекстні, але більшість чисто синтаксичні. Алгоритм Хобса працює за допомогою правил обходу синтаксичного дерева тексту. Центруючий алгоритм оснований на ідеї, що в будь-який момент в тексті є те, на чому зосереджена увага, таким чином потрібно тільки постійно слідкувати за змінами акцентів у тексті.

Література

1. Jurafsky D. Speech and Language Processing / D. Jurafsky, J. Martin.
2. Mayer J. Using the WordNet Hierarchy for Associative Anaphora Resolution / J. Mayer, R. Dale.
3. Hobbs J. Resolving Pronoun References / Hobbs J.
4. Mitkov R. A new, fully automatic version of Mitkov's knowledge-poor pronoun resolution method, 2002 [Електронний ресурс] / Mitkov R., Evans R., Orasan C. – Режим доступу : http://clg.wlv.ac.uk/papers/ciclingAR19.pdf

R.I. Lupiychuk

Anaphora Resolution: Actual Algorithms

Algorithms of anaphora resolution are presented in this paper. The considered algorithms are examined on concrete example. The following algorithms are presented: Lappin & Leass', Hobbs', Centering and Mitkov's.

Стаття надійшла до редакції 01.06.2010.